A Mini Project Report on

SYMPTRACK – A MACHINE LEARNING APPROACH FOR PREDICTING HEART DISEASE AND DIABETES

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

CSE (DATA SCIENCE)

By

POTLA BHAVYA	(21AG1A6754)
LODI ABHIRAM	(21AG1A6738)
BANDHARAM THARUN GOUD	(21AG1A6761)
TIRUMALA SRIVATHSA	(21AG1A6762)

Under the guidance of

Mr. M. Hari Krishna
Assistant Professor

DEPARTMENT OF CSE (DATA SCIENCE)

ACE Engineering College

Ankushapur(V), Ghatkesar(M), Medchal Dist - 501 301

(An Autonomous Institution, Affiliated to JNTUH, Hyderabad)

www.aceec.ac.in

2024-2025

ACE
Engineering College
UGC AUTONOMOUS INSTITUTION
(Sponsored by Yadala Satyanarayana Memorial Educational Society, Hyderabad)
Approved by AICTE & Affiliated to JNTUH
B.Tech Courses offered: CIVIL, CSE, IT, ECE, EEE & MECH, NBA Accredited Courses: CIVIL, CSE, ECE, EEE & MECH, Accorded NAAC A - Grade

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the mini project report entitled "**Symptrack – A Machine Learning Approach for Predicting Heart Disease and Diabetes**" is a Bonafide work done by ***Potla Bhavya (21AG1A6754), Lodi Abhiram (21AG1A6738), Bandharam Tharun Goud (21AG1A6761), Tirumala Srivathsa (21AG1A6762),*** in partial fulfillment for the award of Degree of BACHELOR OF TECHNOLOGY in ***CSE (Data Science)*** from JNTUH University, Hyderabad during the academic year 2024- 2025. This record of Bonafide work carried out by them under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma.

Mr. M. Hari Krishna	**Dr. P. Chiranjeevi**	**External**
Assistant Professor	Associate Professor	
Supervisor	H.O.D, CSE-DS	

ACKNOWLEDGEMENTS

we would like to express our gratitude to all the people behind the screen who have helped our transform an idea into a real time application.

we would like to express our heart-felt gratitude to our parents without whom we would not have been privileged to achieve and fulfill our dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy,** for having founded such an esteemed institution. Sincere thanks to our Joint Secretary **Mrs. M. Padmavathi**, for support in doing project work. we are also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project.

we profoundly thank **Dr. P. Chiranjeevi**, Associate Professor and Head of the Department of Computer Science and Engineering (Data Science), who has been an excellent guide and also a great source of inspiration to our work.

we extremely thank **Mr. P. Ashok Kumar and Mr. Shaik Nagur Vali,** Associate Professors, Project coordinators, who helped us in all the way in fulfilling of all aspects in completion of our Major-Project.

we are very thankful to my internal guide **Mr. M. Hari Krishna** who has been an excellent and also given continuous support for the Completion of our project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, we would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

P. Bhavya	**(21AG1A6754)**
L. Abhiram	**(21AG1A6738)**
B. Tharun	**(21AG1A6761)**
T. Srivathsa	**(21AG1A6762)**

SYMPTRACK – A MACHINE LEARNING APPROACH FOR PREDICTING HEART DISEASE AND DIABETES

ABSTRACT

This study explores the application of machine learning (ML) to predict multiple diseases, with a primary focus on diabetes. Leveraging patient data such as age, lifestyle habits, medical history, and laboratory results, we aimed to build an efficient predictive model for early diagnosis. Various ML algorithms, including Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM) were implemented and tested for accuracy and reliability.

Among the models, Random Forest emerged as the top performer, demonstrating exceptional accuracy in predicting diabetes. Furthermore, the model showed potential for extending its capabilities to predict other conditions like hypertension. This underscores its adaptability and effectiveness in addressing broader healthcare challenges.

The system is implemented as a user-friendly web application designed to assist healthcare professionals and patients. It simplifies the process of early detection, enabling timely interventions and effective health management. With continuous monitoring and regular updates, the system ensures it remains accurate and relevant in dynamic healthcare environments.

This initiative highlights the transformative potential of ML in healthcare, paving the way for proactive and personalized patient care. By utilizing advanced technology, we contribute to improving health outcomes and enhancing the efficiency of healthcare systems.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Heart disease and diabetes are two of the most pressing global health challenges, causing significant mortality and morbidity. Heart disease, including coronary artery disease, is a leading cause of death, while diabetes, a chronic metabolic disorder, can lead to severe complications such as kidney failure, cardiovascular disease, and neuropathy. Early detection and timely management of these conditions are crucial to reducing their impact on individuals and healthcare systems.

Machine learning (ML) has emerged as a transformative tool in addressing these challenges. ML models can process large, complex datasets to identify patterns and relationships, enabling accurate predictions and proactive healthcare solutions. The process starts with **data collection**, where historical patient data, such as medical records, lifestyle factors, and laboratory results, is gathered. **Data preprocessing** ensures the datasets are cleaned, normalized, and prepared for analysis, handling inconsistencies and missing values. **Feature selection** identifies relevant predictors, such as glucose levels, cholesterol, age, and family history, to enhance model performance.

Advanced **ML models** like Random Forest, Neural Networks, and Support Vector Machines are trained on the pre-processed data to learn patterns specific to heart disease and diabetes. Rigorous **evaluation** methods, such as cross-validation and metrics like Mean Squared Error (MSE) or accuracy, ensure the model's reliability. **Hyperparameter tuning** further optimizes model performance for real-world applicability.

Once trained and validated, these models are deployed in a user-friendly system, such as a web application, allowing healthcare professionals to make real-time predictions. This enables early diagnosis, risk assessment, and personalized treatment plans. Continuous updates ensure that the system adapts to new data and remains relevant in dynamic healthcare environments.

By integrating ML into healthcare, this approach significantly enhances diagnostic accuracy, reduces costs, and improves overall patient outcomes, making it a valuable tool in combating heart disease and diabetes.

1. Data Collection

Data collection involves gathering historical health data, including patient demographics, medical histories, lab results, and lifestyle factors. This comprehensive dataset serves as the foundation for building machine learning models. External factors like socioeconomic status or geographic trends can also be included. A high-quality dataset is crucial to ensure accurate predictions.

2. Data Preprocessing

Data preprocessing is essential for ensuring the quality and consistency of the collected dataset. It involves cleaning data to handle missing values, removing outliers, and normalizing attributes to maintain uniformity. This step also includes organizing the data into structured formats for analysis. Proper preprocessing minimizes noise, ensuring that models focus on meaningful patterns. It lays the groundwork for effective and reliable machine learning models.

3. Feature Selection

Feature selection identifies the most relevant attributes contributing to disease prediction. For heart disease and diabetes, features like glucose levels, cholesterol, age, and blood pressure are critical. This process reduces the dimensionality of the dataset, improving computational efficiency. Selected features enhance the interpretability of the model and its performance. Techniques like correlation analysis and domain knowledge guide this selection process.

4. Model Selection

Choosing the right machine learning algorithm is vital for effective predictions. Models like Random Forest, Support Vector Machines, Neural Networks, and Decision

Trees are evaluated for their suitability. The choice depends on the complexity of the problem and the nature of the dataset. Factors such as accuracy, interpretability, and computational efficiency are considered. Ensemble methods, like Gradient Boosting, can also be used to combine model strengths.

5. Training the Model

Model training involves using the pre-processed dataset to teach the algorithm the underlying patterns. The dataset is split into training and testing subsets to evaluate the model's ability to generalize. During training, the algorithm learns relationships specific to predicting heart disease and diabetes. The process often requires multiple iterations to optimize the learning process. This step ensures the model can make accurate predictions based on unseen data.

6. Evaluation

Evaluation measures the model's performance in predicting outcomes accurately. Metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy scores are used. Validation techniques, such as cross-validation, ensure the model generalizes well to new data. Residual analysis helps identify where the model may be underperforming. Effective evaluation ensures the reliability of the predictive system in real-world applications.

7. Hyperparameter Tuning

Hyperparameter tuning optimizes the model's parameters to improve its performance. Techniques like grid search and random search are used to explore combinations of hyperparameters. Cross-validation ensures that the tuned parameters enhance model robustness. Proper tuning minimizes overfitting and underfitting issues, improving generalization. This iterative process is critical for achieving a balance between model complexity and performance.

8. Prediction

Once trained and evaluated, the model is used to predict the likelihood of diseases in new data. Predictions can guide healthcare professionals in early diagnosis and intervention. Results are typically presented as probabilities, allowing for risk assessment. Continuous monitoring ensures the predictions remain reliable as new data becomes available. This step bridges the gap between model development and practical application.

9. Deployment

Deployment integrates the trained model into a user-friendly system, such as a web application. Healthcare professionals can input patient data to receive real-time predictions. Deployment includes creating an interface that is accessible and interpretable by non-technical users. Regular updates ensure the model adapts to new data and changing healthcare scenarios. This makes the predictive system a valuable tool for decision-making in clinical settings.

2. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical step in understanding the structure, patterns, and relationships within a dataset. It involves summarizing data through visualizations, statistical measures, and initial insights. For healthcare-related datasets, such as those predicting heart disease and diabetes, EDA helps uncover trends in features like age, glucose levels, blood pressure, and cholesterol.

The process starts with visualizing data distributions using histograms, box plots, or scatter plots to identify any skewness, outliers, or anomalies. Correlation matrices reveal relationships between variables, such as the link between glucose levels and diabetes or cholesterol and heart disease. Such insights guide feature selection and model building.

EDA also helps detect missing data, outliers, and inconsistencies that may impact model performance. Techniques like imputation or transformation are used to handle these issues. Additionally, decomposing time-series data into trends, seasonality, and residuals provides a deeper understanding of temporal patterns, especially in stock or healthcare datasets.

By providing a foundation for data preprocessing and feature selection, EDA ensures that machine learning models are built on reliable and meaningful data. It is an essential step that bridges raw data and actionable insights, enabling more robust and accurate predictive systems.

1. Data Collection:

Data collection involves gathering relevant datasets from diverse sources, such as patient records, lab results, and public health databases. For heart disease and diabetes prediction, this includes information on age, glucose levels, cholesterol, blood pressure, and lifestyle habits. Ensuring data completeness and relevance is critical at this stage. External data, such as socioeconomic indicators and environmental factors, may also be included for deeper analysis.

2. Data Inspection:

Data inspection focuses on assessing the quality and structure of the collected dataset. This involves checking for missing values, identifying outliers, and verifying the consistency of data entries. Statistical summaries and visualizations, such as histograms and scatter plots, are used to detect anomalies and trends. This step ensures the dataset aligns with the requirements of machine learning models.

3. Feature Engineering:

Feature engineering transforms raw data into meaningful inputs for predictive models. New features, such as Body Mass Index (BMI) or risk scores, can be derived from existing data. Additionally, technical indicators, such as moving averages, are generated for temporal datasets like stock prices. Feature engineering enhances the dataset's predictive power, ensuring relevant information is highlighted.

4. Data Preprocessing:

Data preprocessing ensures the dataset is ready for analysis by handling missing values, outliers, and inconsistencies. This includes scaling features to a uniform range, encoding categorical variables, and normalizing continuous attributes. Preprocessing ensures data uniformity, prevents certain features from dominating the model, and enhances performance.

3. LITERATURE SURVEY

The integration of machine learning (ML) for predicting heart disease and diabetes has revolutionized healthcare diagnostics. Several studies have highlighted the effectiveness of ML in analysing complex medical datasets to identify patterns and provide accurate predictions. These approaches offer a proactive way to manage chronic conditions and reduce healthcare burdens.

Logistic Regression has been widely used for binary classification tasks, such as predicting the presence or absence of heart disease based on features like cholesterol, blood pressure, and age. Similarly, Support Vector Machines (SVM) have proven effective in diabetes prediction, especially when analysing features such as glucose levels and BMI. These traditional algorithms provide robust baselines but often struggle with larger, noisier datasets.

More recent approaches focus on ensemble methods, such as Random Forest and Gradient Boosting. These algorithms outperform traditional methods by combining the strengths of multiple models, providing better accuracy and handling high-dimensional data. Neural networks, particularly deep learning models, have also been applied to learn non-linear relationships in datasets containing diverse features like medical histories, lab results, and demographic details.

Feature selection and engineering are critical for improving model performance. Studies have demonstrated that derived features, such as risk scores or combined factors like BMI and cholesterol levels, significantly enhance predictive accuracy. Moreover, advanced preprocessing methods, including normalization, scaling, and handling missing values, ensure model robustness.

Time-series analysis is particularly relevant in healthcare, enabling sequential predictions for monitoring disease progression. Techniques like Long Short-Term Memory (LSTM) networks have been employed to capture temporal dependencies in patient data, offering better insights for dynamic risk assessment.

4. SYSTEM REQUIREMENT ANALYSIS

4.1. EXISTING SYSTEM

Traditional healthcare systems rely heavily on manual methods and clinical expertise to predict and diagnose diseases like heart disease and diabetes. These approaches include statistical analyses and conventional algorithms that often focus on individual disease-specific datasets. While these methods have been useful, they come with significant limitations in terms of accuracy, scalability, and adaptability.

For heart disease prediction, Logistic Regression is one of the most commonly used techniques. It estimates the probability of disease presence based on features such as cholesterol, blood pressure, and age. However, this approach often oversimplifies complex relationships between multiple features and is prone to lower accuracy when dealing with larger or noisier datasets.

In diabetes prediction, Support Vector Machines (SVM) are widely applied. SVMs classify patients into diabetic or non-diabetic categories based on features like glucose levels, BMI, and age. Although effective for linear separable data, SVMs struggle with non-linear data patterns or datasets containing high levels of noise and missing values.

Existing systems also lack integration, as they often focus on predicting a single disease at a time. This limits their ability to address comorbidities, such as the frequent coexistence of heart disease and diabetes. Additionally, these systems rarely incorporate real-time data analysis or adapt dynamically to new data trends.

Another drawback is the minimal use of advanced feature selection and engineering techniques, which reduces the overall predictive power of traditional systems. Furthermore, existing solutions are not always user-friendly and require specialized knowledge, making them inaccessible to a broader range of healthcare professionals and patients.

Traditional Approaches:

1. Rule-Based Systems

Traditional diagnostic systems depend on predefined rules and thresholds to assess the risk of diseases. For example, a blood glucose level above a certain threshold indicates diabetes, while elevated cholesterol levels suggest a risk of heart disease. Although these systems are straightforward and interpretable, they do not capture the complex relationships between multiple risk factors.

2. Statistical Models

Logistic Regression is a common statistical approach used for binary classification problems, such as predicting whether a patient has heart disease or diabetes. It models the relationship between independent variables (e.g., glucose levels, cholesterol, and blood pressure) and a binary dependent variable. While effective for simple relationships, Logistic Regression assumes linearity and struggles to capture non-linear interactions.

3. Clinical Risk Scores

Risk scoring systems, like the Framingham Risk Score for heart disease, estimate the probability of developing a condition based on a few key features, such as age, smoking status, and cholesterol levels. While these scores are easy to use in clinical settings, they do not incorporate advanced data analysis techniques or adapt to individual variations.

Support Vector Machine (SVM) Algorithm

The SVM algorithm is one of the most popular machine learning techniques used in existing systems for disease prediction. SVM works by finding an optimal hyperplane that separates data points into distinct classes, such as "disease" and "no disease."

1. Application in Heart Disease

SVM is used to classify patients based on features like cholesterol, blood pressure, and age. The algorithm handles both linearly and non-linearly separable data using kernel functions. For heart disease prediction, radial basis function (RBF) kernels are often applied to capture non-linear relationships.

2. Application in Diabetes

In diabetes prediction, SVM uses features like glucose levels, BMI, and family history to classify patients as diabetic or non-diabetic. It is particularly effective in high-dimensional spaces, making it a good fit for datasets with multiple features.

3. Strengths of SVM

- SVM performs well with small to medium-sized datasets.
- It is effective at handling non-linear data when paired with kernel functions.
- The algorithm provides robust generalization, reducing the risk of overfitting.

4. Limitations of SVM

- Computational inefficiency makes SVM challenging to use with very large datasets.
- It is sensitive to the choice of hyperparameters, such as the kernel type and regularization parameter.
- Interpretability is limited compared to simpler models like Logistic Regression or Decision Trees.

Basic Machine Learning Models

1. Decision Trees

Decision Trees classify patients based on a series of questions, such as "Is glucose > 120?" or "Is cholesterol > 200?" While intuitive and interpretable, Decision Trees are prone to overfitting, which can reduce their generalization to new data.

2. K-Nearest Neighbors (KNN)

KNN algorithms classify patients by comparing them to their "k" nearest neighbors in the dataset. For example, the classification of a patient as diabetic or non-diabetic depends on the outcomes of similar patients in terms of glucose and BMI. However, KNN suffers from scalability issues with large datasets and is sensitive to irrelevant features.

Limitations of Existing System

1. Limited Feature Integration

Most existing systems focus on a small set of features, such as glucose and cholesterol levels, ignoring other important factors like lifestyle, socioeconomic status, and real-time health data.

2. Static Models

Existing models are static and do not adapt to new data or evolving trends in patient health and medical research. This limits their applicability in dynamic healthcare environments.

3. Scalability Issues

Basic models, including SVM, face challenges when applied to large or high-dimensional datasets, which are common in healthcare.

4. Interpretability

While some models, like Logistic Regression, are interpretable, others, including SVM, lack transparency in their decision-making process, making them less suitable for clinical adoption.

Examples of Existing Systems in Practice

1. Diabetes Prediction

SVM is commonly used for classifying patients as diabetic or non-diabetic based on features like glucose levels, BMI, and age. These systems provide reasonable accuracy but struggle with scalability and interpretability.

2. Heart Disease Prediction

SVM is also employed for heart disease prediction, utilizing features such as cholesterol, blood pressure, and age. While the algorithm effectively handles non-linear relationships, its computational cost limits its usability in large-scale applications.

4.2. PROPOSED SYSTEM

The proposed model aims to enhance the prediction of heart disease and diabetes by addressing the limitations of existing systems and integrating advanced machine learning (ML) techniques. It focuses on building a comprehensive, accurate, and user-friendly system for real-time disease prediction and management. This model leverages shared risk factors for heart disease and diabetes, such as cholesterol levels, age, and family history, to create a unified predictive solution.

Key Aspects:

1. Overview of Random Forest

Random Forest operates by constructing a multitude of decision trees during training and combining their outputs to make a final prediction. It falls under the category of ensemble learning methods, where multiple models are trained to solve the same problem, and their results are aggregated for a more robust and accurate prediction.

In classification tasks, Random Forest predicts the class that receives the majority vote from all the decision trees. For regression tasks, it outputs the average of all the predictions from individual trees. This aggregation makes Random Forest resilient to overfitting and highly reliable in diverse applications.

2. Ensemble Learning

The core principle of Random Forest lies in ensemble learning. Ensemble learning enhances the performance of weak models (in this case, individual decision trees) by combining their predictions. While individual decision trees may have high variance or overfit the data, Random Forest mitigates this by aggregating the predictions, leading to better generalization.

3. Random Sampling with Bagging

Random Forest employs a technique known as bagging (Bootstrap Aggregating) to train the decision trees. In bagging, multiple datasets are generated by sampling the original data with replacement.

4. Random Feature Selection

Another critical feature of Random Forest is its use of random subsets of features at each split in the decision trees. Unlike traditional decision trees, which evaluate all features to determine the best split, Random Forest only considers a random subset of features.

This feature randomness ensures that the trees are diverse and reduces the chances of any dominant feature influencing the model disproportionately. It also enhances the algorithm's ability to handle high-dimensional data effectively.

5. Robustness and Accuracy

Random Forest is known for its high accuracy and robustness across a wide range of datasets. It performs exceptionally well with large datasets, noisy data, and high-dimensional features. The algorithm's ability to average multiple predictions makes it less sensitive to noise and outliers.

Random Forest often outperforms simpler models like logistic regression or individual decision trees, particularly when the relationships between features and the target variable are complex and non-linear.

6. Overfitting Prevention

One of the major advantages of Random Forest is its resilience against overfitting. While individual decision trees can overfit the training data, Random Forest reduces this risk by averaging the predictions of multiple trees.

The randomness introduced in both sampling and feature selection ensures that the trees do not memorize the training data but instead generalize patterns that can be applied to unseen data. This makes Random Forest highly reliable for real-world applications.

7. Feature Importance

Random Forest provides an inherent mechanism to measure feature importance. During training, it evaluates how much each feature contributes to reducing the impurity (e.g., Gini impurity or information gain) in the decision trees.

The algorithm assigns an importance score to each feature, which helps in identifying the most influential variables in the dataset. This is particularly useful for feature selection and understanding the dataset's structure.

8. Versatility

Random Forest is a versatile algorithm capable of handling both categorical and numerical data. It can be applied to a wide range of problems, including classification, regression, and unsupervised learning tasks like clustering and anomaly detection.

For classification tasks, Random Forest predicts the majority class, while for regression tasks, it averages the predicted values of all trees. This flexibility makes it a popular choice in fields like healthcare, finance, and e-commerce.

9. Scalability

Random Forest is scalable to large datasets due to its parallelizable structure. Each decision tree in the forest is built independently, allowing for parallel processing. This significantly reduces the training time when implemented on distributed computing frameworks like Apache Spark.

While computationally more intensive than simpler algorithms, Random Forest balances accuracy and scalability effectively, making it suitable for modern, large-scale datasets.

10. Handling Missing Data

Random Forest has built-in mechanisms to handle missing data. It can use surrogate splits or impute missing values during training, ensuring that incomplete datasets do not compromise the model's performance.

11. Interpretability

While Random Forest is generally considered a black-box model compared to simpler algorithms like Logistic Regression, it offers interpretability through feature importance scores. By analyzing these scores, practitioners can gain insights into which features drive the predictions.

For example, in a stock market prediction scenario, Random Forest can reveal that features like trading volume and historical price changes are most influential in predicting future stock prices.

12. Hyperparameter Tuning

Random Forest includes several hyperparameters that can be tuned to optimize its performance. **Key hyperparameters include:**

- **n_estimators**: The number of decision trees in the forest. More trees generally improve performance but increase computational cost.
- **max_depth:** The maximum depth of each tree, controlling overfitting and underfitting.
- **max_features:** The number of features considered for splitting at each node, balancing diversity and accuracy.
- **min_samples_split and min_samples_leaf**: Minimum samples required to split a node or form a leaf, reducing overfitting.
- Hyperparameter tuning can be done using techniques like grid search or random search, often combined with cross-validation to ensure robust model performance.

13. Advantages of Random Forest

- **High Accuracy:** It often outperforms simpler models due to its ability to capture complex patterns.
- **Robustness:** Handles noisy and unbalanced datasets effectively.
- **Feature Importance:** Provides insights into which features contribute most to predictions.
- **Scalability:** Can handle large datasets and high-dimensional data efficiently.
- **Versatility:** Suitable for both classification and regression tasks.

14. Limitations of Random Forest

- **Computational Cost:** Training a large number of trees can be computationally expensive.
- **Memory Usage:** Requires more memory to store multiple trees compared to single models.
- **Black-Box Nature**: While feature importance provides some interpretability, it is less transparent compared to simpler models like Decision Trees.

Sensitivity to Hyperparameters: Requires careful tuning of hyperparameters for optimal performance.

15. Applications of Random Forest

- **Healthcare:** Predicting diseases like heart disease and diabetes.
- **Finance**: Credit scoring, fraud detection, and stock price prediction.
- **Environment**: Weather forecasting and detecting deforestation patterns.

4.3. SOFTWARE REQUIREMENT SPECIFICATIONS

Operating System	: Linux/Windows 10
Language	: Python
Front-End	: Python
IDE	: Visual Studio Code
Libraries used	: Streamlit, NumPy, Pandas, sklearn
Designing	: Streamlit

4.4. HARDWARE REQUIREMENT SPECIFICATIONS

Processor	:	AMD Ryzen™ 3 / Intel Core i3 or above
Memory	:	4GB
Hard Disk	:	256 GB SSD

4.5. SYSTEM ANALYSIS

Interaction Module

The Interaction Module is the interface that connects healthcare professionals to the system, allowing them to input patient data and view prediction results effortlessly.

Key Components:

1. Input Entry:

- Facilitates the entry of patient health data, such as glucose levels, blood pressure, BMI, and age, in a structured manner.
- Ensures the validity of data through dropdown menus, radio buttons, or predefined ranges, reducing the risk of input errors.

2. User Interface (Streamlit):

- A user-friendly web interface designed to provide intuitive navigation and interactivity.
- Streamlit offers real-time feedback and updates, making it suitable for quick data entry and result display.

3. View Results:

- Displays the prediction results in an easy-to-understand format, like “Diabetes Detected” or “Low Risk of Heart Disease.”
- Includes visual aids such as graphs or charts to highlight patterns and trends in patient data, enabling better decision-making.

Model Saving Module

The Model Saving Module ensures that the machine learning model can be reused without retraining, saving computational resources and time.

Key Components:

1. **Saving the Model:**
   - The trained model is serialized using a reliable mechanism to store it in a portable format.
   - This allows healthcare professionals to use the same model for consistent predictions over multiple sessions or deployments.
2. **Loading the Model:**
   - When predictions are required, the saved model is deserialized and loaded into memory.
   - This ensures that the system can provide results instantly without re-training the model.
3. **Benefits:**
   - Reduces system overhead and accelerates response times.
   - Allows easy sharing of the model between different teams or systems.

Prediction Module

The Prediction Module analyzes the patient data provided via the Interaction Module to generate predictions about diabetes and heart disease risks.

Key Components:

1. **Data Collection:**
   - Gathers all necessary patient data in a structured format, ready for analysis.
   - Ensures that the data adheres to the required format for compatibility with the machine learning model.

2. **Machine Learning Model:**
   - Processes the input features (e.g., glucose levels, blood pressure) to classify the likelihood of diabetes and heart disease.
   - Outputs a clear result, such as a percentage risk or a binary decision (Yes/No).

3. **Result Generation:**
   - Provides predictions with confidence levels, allowing healthcare professionals to gauge the reliability of the results.
   - Highlights key factors influencing the prediction (e.g., high glucose levels contributing to diabetes risk).

4. **Future Potential:**
   - The module can be extended to support predictions for other diseases or integrate with IoT devices for real-time health monitoring.

Data Preprocessing Module

The Data Preprocessing Module ensures the raw input data is transformed into a clean and structured format suitable for the machine learning model.

Key Components:

1. **Data Cleaning:**
    - Identifies and resolves missing, incomplete, or erroneous data entries.
    - Removes irrelevant data or duplicates to enhance accuracy.
2. **Feature Engineering:**
    - Creates derived features like BMI from height and weight to enhance model input.
    - Selects only the most relevant features to reduce complexity and noise.
3. **Standardization and Normalization:**
    - Ensures that numerical data is on the same scale, improving the performance of the ML model.
4. **Class Balancing:**
    - Addresses imbalanced datasets by augmenting minority class data or under sampling the majority class.

Model Training Module

This module focuses on building a robust and accurate machine learning model using the preprocessed data.

Key Components:

1. **Algorithm Selection:**
    - Utilizes the Random Forest algorithm, known for its high accuracy and ability to handle large datasets effectively.
2. **Dataset Splitting:**
    - Divides the dataset into training and testing sets to ensure the model generalizes well to unseen data.

3. **Hyperparameter Tuning:**
    - Fine-tunes parameters like the number of trees and depth of the forest for optimal performance.
4. **Validation:**
    - Evaluates the model using cross-validation techniques to ensure its reliability across different subsets of data.

Evaluation and Testing Module

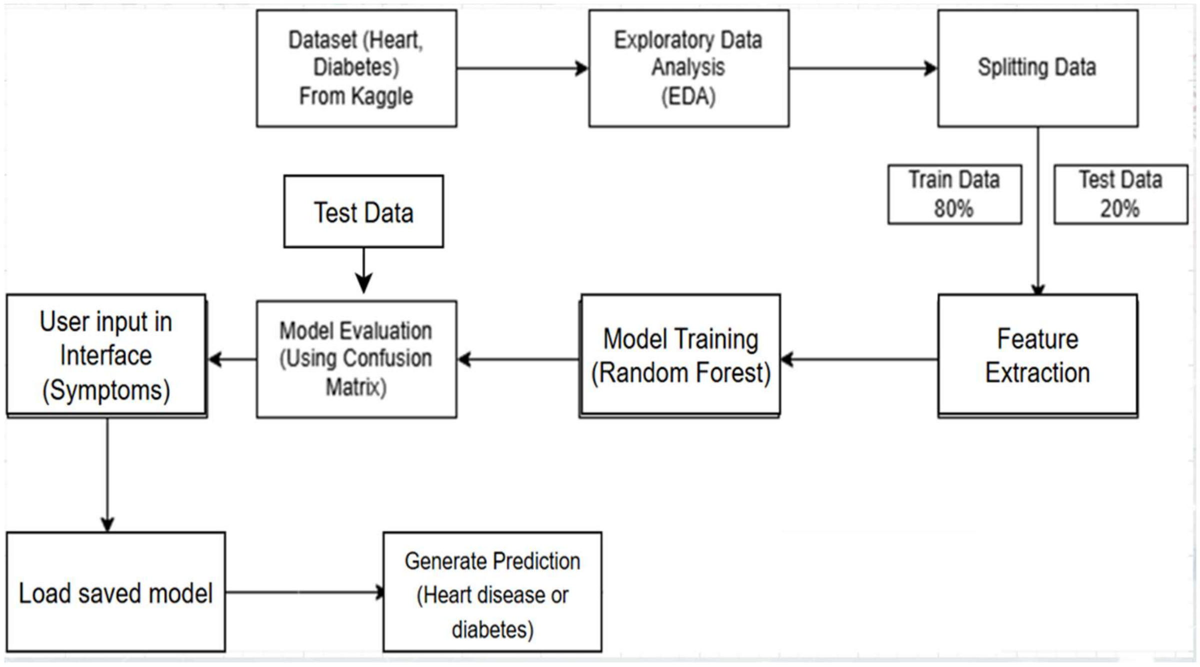
This module measures the performance of the machine learning model to ensure its predictions are accurate and actionable.

Key Components:

1. **Performance Metrics:**
    - Includes accuracy, precision, recall, and F1-score to assess classification quality.
2. **Error Analysis:**
    - Identifies common misclassifications and adjusts the model to minimize these errors.
3. **Confusion Matrix:**
    - Visualizes the results of predictions, showing true positives, false positives, true negatives, and false negatives.
4. **Real-World Testing:**
    - Validates the model on real-world datasets to ensure its applicability in practical healthcare setting

5. SYSTEM DESIGN

5.1. PIPELINE ARCHITECTURE


Dataset (Heart, Diabetes) From Kaggle
Exploratory Data Analysis (EDA)
Splitting Data
Train Data 80%
Test Data 20%
Test Data
User input in Interface (Symptoms)
Model Evaluation (Using Confusion Matrix)
Model Training (Random Forest)
Feature Extraction
Load saved model
Generate Prediction (Heart disease or diabetes)

1. Dataset (Heart, Diabetes)

The dataset is the backbone of any machine learning pipeline, providing the essential information for prediction. In this pipeline:

Sources:

- The datasets for heart disease and diabetes are sourced from Kaggle or other reputable platforms. These datasets typically include hundreds or thousands of samples, each with multiple features that influence the diseases.

Features for Heart Disease:

- Common attributes might include age, gender, chest pain type, blood pressure, cholesterol, fasting blood sugar, electrocardiogram results, maximum heart rate, and more.

Features for Diabetes:

- Attributes typically include glucose concentration, BMI, blood pressure, insulin levels, and pregnancy history.

Data Quality:

- Missing values are common and must be managed during preprocessing.
- Balancing the dataset to ensure equal representation of cases with and without diseases is crucial to prevent bias in the machine learning model.

2. Exploratory Data Analysis (EDA)

EDA is a critical step for understanding and preparing the data for further processing. In this pipeline:

Objective:

- Identify patterns and trends, such as which features are most correlated with heart disease or diabetes.
- Understand data distributions and detect outliers that may skew predictions.
- Determine whether any data transformation (e.g., scaling or normalization) is necessary.

Techniques and Tools:

- Univariate Analysis: Analyze single features, such as age distribution or average cholesterol levels.
- Bivariate and Multivariate Analysis: Explore relationships between features (e.g., the correlation between blood pressure and diabetes risk).

Visualization:

- Histograms, box plots, and scatter plots to observe feature distributions.
- Heatmaps to visualize feature correlations.

Outcomes:

- Feature importance insights to guide the next steps in the pipeline.
- Cleaning strategies for missing or inconsistent data.

3. Splitting Data

The dataset must be split to enable proper training and testing of the machine learning model.

Split Ratios:

- Training Data: 80% of the dataset is used to teach the model, identifying patterns and building predictive capabilities.
- Testing Data: 20% of the dataset remains unseen during training and is used solely to evaluate the model.

Why Splitting Matters:

- Training data builds the model, while testing data evaluates it in a real-world context.
- This separation ensures that the model generalizes well and doesn't overfit.

Implementation:

- Using Scikit-learn's train_test_split function, which randomly divides the data while ensuring each class is proportionally represented in both sets (if stratification is required).

Cross-Validation:

- Employ techniques like k-fold cross-validation to further evaluate the model's performance across multiple subsets of data.

4. Feature Extraction

Feature extraction focuses on isolating the most impactful attributes that influence predictions.

Why Feature Extraction?

- Reduces data complexity while maintaining essential information.
- Enhances model accuracy by removing irrelevant or redundant features.

Techniques:

- Correlation Analysis: Features highly correlated with the target variable are retained.

Feature Importance from Random Forest:

- After training, Random Forest models provide importance scores for each feature.
- Features with low importance scores are discarded.

Dimensionality Reduction Techniques:

- Algorithms like PCA (Principal Component Analysis) may be used when the dataset has too many features.

Tools:

- Pandas for manual feature selection.
- Scikit-learn's feature selection utilities for automated extraction.

5. Model Training (Random Forest):

This step involves building the heart of the predictive system—the machine learning model.

Algorithm Used:

- Random Forest, a versatile ensemble learning technique, combines multiple decision trees to provide accurate and robust predictions.

Advantages of Random Forest:

- Handles both numerical and categorical data seamlessly.
- Provides feature importance insights.
- Is resistant to overfitting due to ensemble averaging.
- Performs well on imbalanced datasets.

Training Process:

- Model is trained using the training data subset.
- The hyperparameters, such as the number of trees and tree depth, are fine-tuned for maximum performance.

Implementation Tools:

- Scikit-learn's RandomForestClassifier is used to implement and train the model.

6. Model Evaluation (Confusion Matrix):

Evaluating the trained model is essential for assessing its effectiveness.

Confusion Matrix Explained:

- True Positives (TP): Correctly predicted positive cases (e.g., correctly identified disease cases).
- True Negatives (TN): Correctly predicted negative cases (e.g., no disease cases predicted as such).
- False Positives (FP): Cases incorrectly identified as positive (e.g., healthy patients labeled at risk).
- False Negatives (FN): Cases incorrectly identified as negative (e.g., disease cases missed by the model).

Performance Metrics:

- Accuracy: Overall correctness of the model.
- Precision: Focus on minimizing false positives.
- Recall: Emphasis on minimizing false negatives (critical in healthcare).
- F1-Score: Balances precision and recall into a single metric.
- ROC Curve and AUC: Measures the model's ability to distinguish between classes.

Objective:

- Achieve high recall without significantly sacrificing precision, ensuring fewer disease cases go undetected.

7. User Input in Interface (Symptoms):

The interface collects data directly from the user, typically a healthcare professional or the patient themselves.

Interface Design:

- Simple, intuitive input forms for entering symptoms and clinical data.
- Examples include dropdowns, checkboxes, or manual entry fields for age, glucose level, blood pressure, etc.

Data Collection:

- Ensures proper validation checks to prevent erroneous inputs (e.g., non-numeric characters for age).
- Accepts a standardized format compatible with the trained ML model.

Tools:

- Built using Streamlit, a Python-based library for creating interactive web applications.

8. Load Saved Model:

The trained model, saved using serialization, is retrieved for making real-time predictions.

Serialization Process:

- Pickle is used to store the trained model in a .pkl file.
- Enables loading the model without retraining, saving computational resources.

Benefits of Loading Saved Models:

- Reusability: Models are deployed to production systems quickly.
- Scalability: Saves training time and effort for large-scale implementations.

9. Generate Prediction (Heart disease or Diabetes):

The final step transforms user inputs into actionable predictions.

Prediction Process:

- User inputs are preprocessed and fed into the saved model.
- The model computes the probability of the user being at risk for heart disease or diabetes.
- Thresholds determine the classification (e.g., risk present vs. no risk).

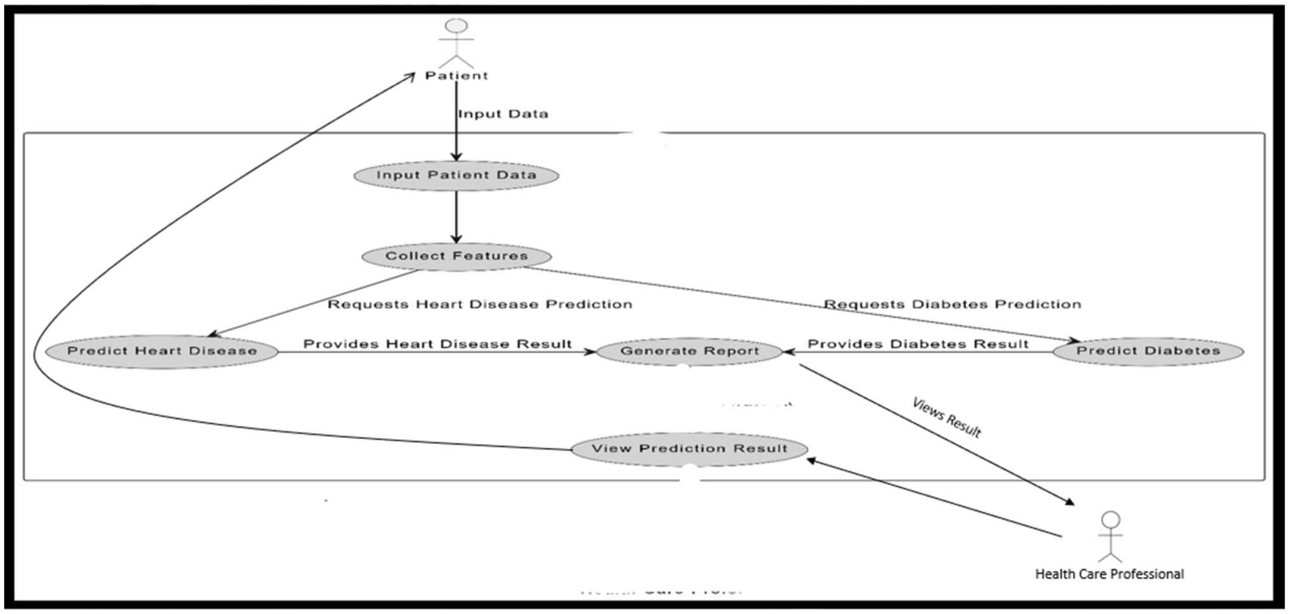
Output:

- Results are displayed in a user-friendly format, such as “Low Risk” or “High Risk.”
- Graphs or statistical scores may be included to explain the results further.

Next Steps:

- Predicted outcomes can be saved or used for generating medical reports.
- These insights can guide diagnostic decisions or preventive actions

5.2. USECASE DIAGRAM



The use case diagram represents the interaction between the system, patients, and healthcare professionals in your project, which predicts heart disease and diabetes using machine learning. It highlights key processes from data input to the generation and viewing of predictions.

1. Actors in the Use Case Diagram:

Patient:

- Primary actor responsible for providing input data.
- Can request predictions for heart disease or diabetes.
- Views the generated prediction results.

Healthcare Professional:

- Secondary actor who accesses prediction results for analysis.
- Uses the results for diagnostic decision-making or preventive strategies.

2. System Processes

Input Patient Data:

- Patients provide information about symptoms, health history, and relevant biomarkers (e.g., glucose level, BMI, age).
- The interface collects and validates this information.

Collect Features:

- The system processes the input data to extract key features required for prediction (e.g., blood pressure for heart disease, glucose levels for diabetes).

Predict Heart Disease:

- When requested by the patient, the system uses the trained machine learning model (Random Forest) to predict the likelihood of heart disease.
- The result is returned as a probability or risk category (e.g., low, medium, high risk).

Predict Diabetes:

- Similarly, the system predicts the likelihood of diabetes when requested by the patient.
- Provides the output with associated risk levels.

Generate Report:

- The system compiles the prediction results into a detailed, easy-to-understand report.
- The report includes diagnostic insights and risk analysis, along with graphical summaries (if implemented).

View Prediction Results:

- Both patients and healthcare professionals can access the prediction outcomes.
- Healthcare professionals can use the results to offer guidance, treatment plans, or further tests.

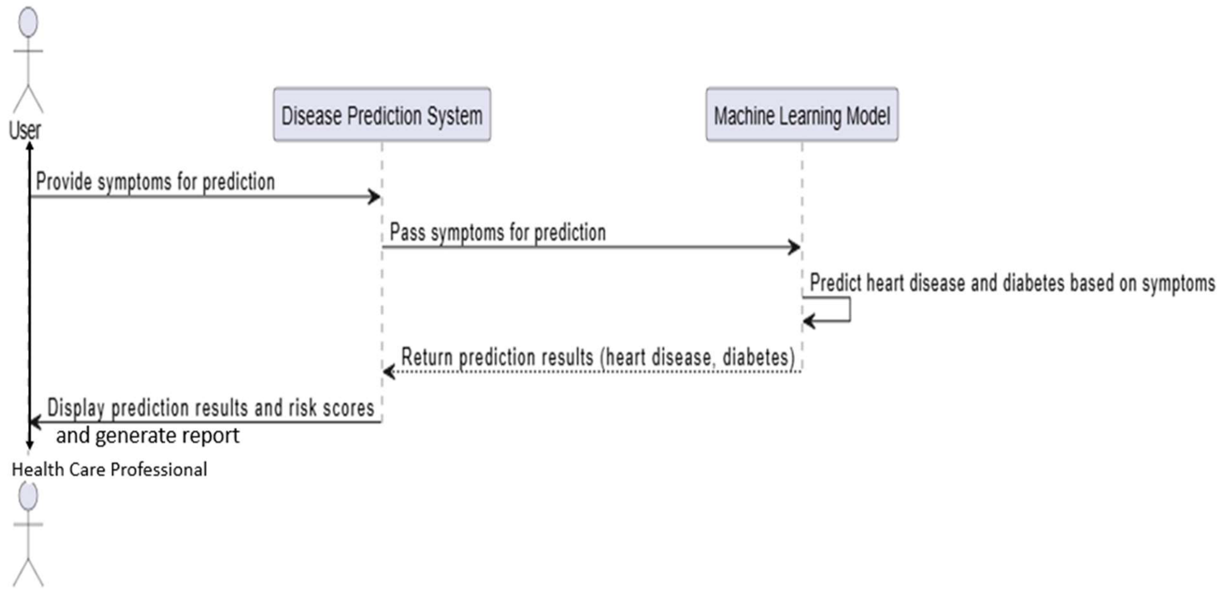
3. Interactions Between Actors and System

- Patients initiate interactions by entering their data and requesting predictions.
- The system processes these requests, provides results, and generates reports accessible to both patients and healthcare professionals.
- Healthcare professionals rely on the results to offer expert advice, improving diagnostic accuracy and medical outcomes.

4. Additional Explanation of Use Case Diagram Flows

- Flow 1: Patients enter their data using a user-friendly interface. The system captures and processes this data in real time.
- Flow 2: Upon a prediction request, the system directs the patient information to the relevant machine learning model (heart disease or diabetes prediction model).
- Flow 3: After computations, the system generates and presents results to patients. Healthcare professionals can access these results via a shared platform or directly in a clinical setting.

5.3. SEQUENCE DIAGRAM


User
Disease Prediction System
Machine Learning Model
Provide symptoms for prediction
Pass symptoms for prediction
Predict heart disease and diabetes based on symptoms
Return prediction results (heart disease, diabetes)
Display prediction results and risk scores and generate report
Health Care Professional

Objective of the Diagram:

The sequence diagram describes how the Disease Prediction System works with healthcare professionals and machine learning models to predict heart disease and diabetes based on patient symptoms. It focuses on user interaction, data processing, model predictions, and result delivery.

Flow of Events:

1. **User Interaction:**
    - The user enters patient symptoms into the system through a user-friendly interface.
    - Example symptoms: glucose level, cholesterol, blood pressure, etc.

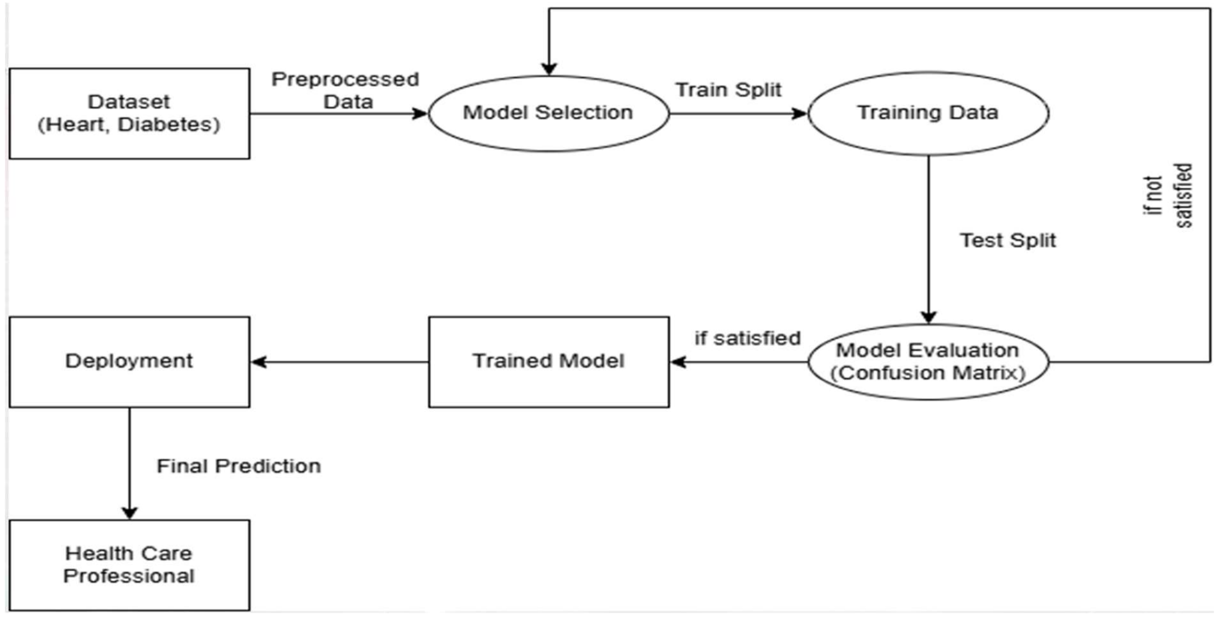
2. **System Processing:**
    - The Disease Prediction System processes the input and sends it to the machine learning model.

3. **Model Prediction:**
    - The machine learning model computes the likelihood of heart disease and diabetes.
4. **Result Presentation:**
    - The Disease Prediction System displays the prediction results and generates a detailed report for healthcare professionals.

Significance

This diagram provides a clear understanding of how data flows through the system, enabling healthcare professionals to leverage machine learning for early detection of critical diseases. It also emphasizes system usability, automation, and efficiency in handling patient data.

5.4. DATA FLOW DIAGRAM



This is a Data Flow Diagram (DFD) that represents the process flow of your project for predicting heart disease and diabetes. Here's an explanation of each step:

1. Dataset (Heart, Diabetes):

- This is the starting point of the project, where the data is collected.
- The dataset includes medical records related to heart disease and diabetes, with features like age, glucose levels, cholesterol levels, and family history.
- The raw data undergoes preprocessing to remove inconsistencies, handle missing values, and normalize/standardize features as required.

2. Preprocessed Data:

- The output of the preprocessing step, which transforms raw data into a clean, consistent format suitable for machine learning.
- This step ensures that all necessary features are correctly scaled and encoded to improve model performance.

3. Model Selection:

- This involves choosing the most appropriate machine learning algorithms for prediction. In your project:
- Logistic Regression is used for interpreting binary outcomes like the presence or absence of disease.
- Random Forest is selected for its robustness and ability to handle nonlinear relationships between variables.

4. Train Split:

- A portion of the preprocessed data is used to train the machine learning models.
- The training data helps the models learn patterns and relationships between features and the target variables (heart disease and diabetes).

5. Training Data:a

- This represents the data fed into the model during the training phase.
- It is used to optimize the model's weights and parameters to minimize prediction error.

6. Test Split:

- The remaining portion of the dataset is reserved for testing the model.
- This ensures the model's performance is evaluated on unseen data to assess generalization ability.

7. Model Evaluation (Confusion Matrix):

- The trained model is evaluated using metrics derived from a confusion matrix, such as accuracy, precision, recall, and F1-score.
- If the performance is unsatisfactory, the process loops back to adjust the model or preprocess the data further.

8. Trained Model:

- The final model obtained after training and satisfactory evaluation.
- It is saved using serialization (e.g., using the Pickle library) for future use and deployment.

9. Deployment:

- The trained model is integrated into a system that healthcare professionals can access.
- This involves creating a user-friendly interface using tools like Streamlit to input patient data and display prediction results.

10. Final Prediction:

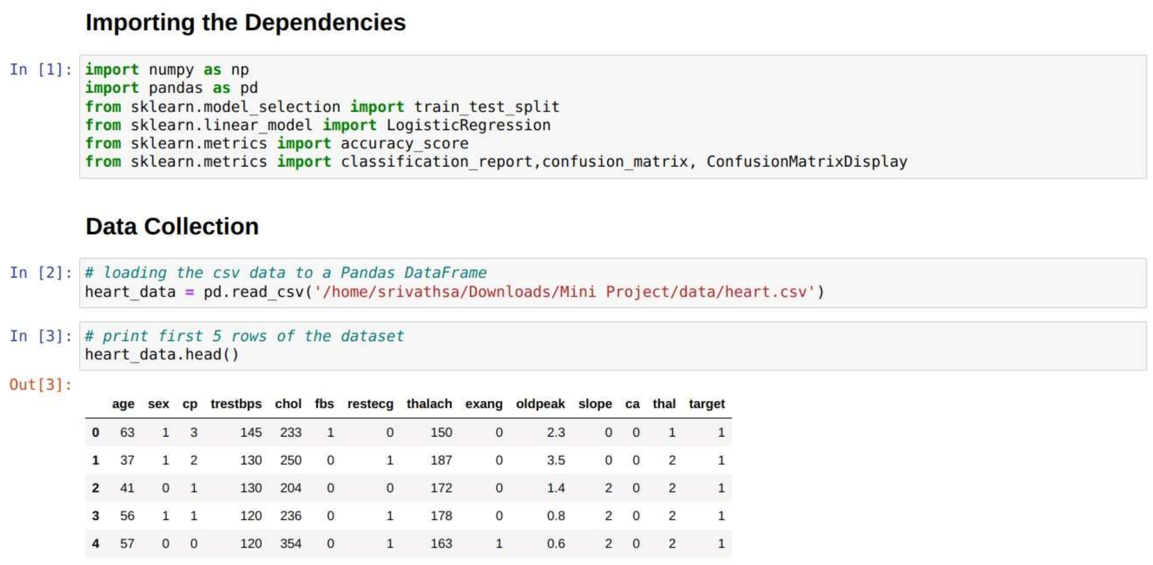
- The deployed system takes user input (patient details) and uses the trained model to make predictions about the likelihood of heart disease or diabetes.
- The results are presented in a clear, actionable format for healthcare professionals.

11. Health Care Professional:

- The end-user who interacts with the deployed system.
- They use the predictions to make informed decisions about patient care, early interventions, and treatment planning.

6. RESULT

6.1 Heart Disease Dataset



Importing the Dependencies

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import classification_report,confusion_matrix, ConfusionMatrixDisplay
```

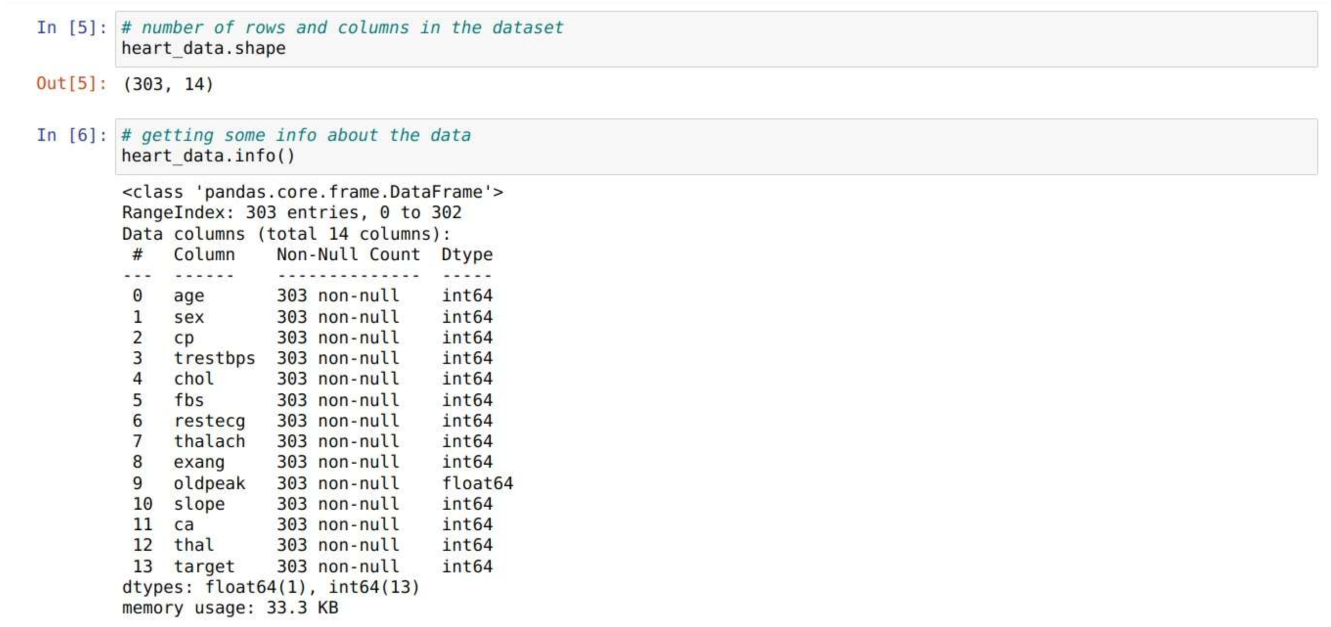
Data Collection

```
In [2]: # loading the csv data to a Pandas DataFrame
        heart_data = pd.read_csv('/home/srivathsa/Downloads/Mini Project/data/heart.csv')
```

```
In [3]: # print first 5 rows of the dataset
        heart_data.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1



```
In [5]: # number of rows and columns in the dataset
        heart_data.shape

Out[5]: (303, 14)

In [6]: # getting some info about the data
        heart_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Splitting the Data into Training data & Test Data

```
In [13]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
In [14]: print(X.shape, X_train.shape, X_test.shape)

         (303, 13) (242, 13) (61, 13)
```

Model Training

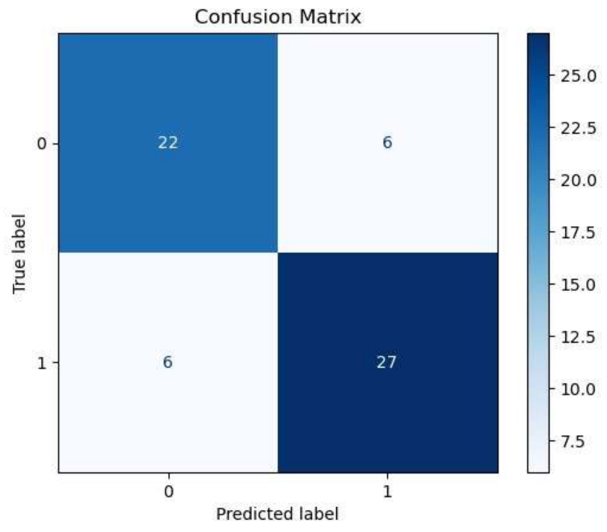
Logistic Regression

```
In [25]: model = LogisticRegression(max_iter=4000)
```

```
In [28]: # training the LogisticRegression model with Training data
         model.fit(X_train, Y_train)

Out[28]: ▸ LogisticRegression
```

Model Evaluation

```
In [20]: from sklearn.metrics import classification_report
         cm = confusion_matrix(Y_test, X_test_prediction)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm)
         disp.plot(cmap='Blues')
         plt.title('Confusion Matrix')
         plt.show()
         print(classification_report(Y_test,X_test_prediction))
```



```
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        28
           1       0.82      0.82      0.82        33

    accuracy                           0.80        61
   macro avg       0.80      0.80      0.80        61
weighted avg       0.80      0.80      0.80        61
```

Saving the trained model

```
In [28]: import pickle
```

```
In [29]: filename = 'heart_disease_model.sav'
         pickle.dump(model, open(filename, 'wb'))
```

```
In [30]: # loading the saved model
         loaded_model = pickle.load(open('heart_disease_model.sav', 'rb'))
```

```
In [31]: for column in X.columns:
           print(column)
```

```
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
```

Building a Predictive System

```
In [22]: input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

         # change the input data to a numpy array
         input_data_as_numpy_array= np.asarray(input_data)

         # reshape the numpy array as we are predicting for only on instance
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

         prediction = model.predict(input_data_reshaped)
         print(prediction)

         if (prediction[0]== 0):
           print('The Person does not have a Heart Disease')
         else:
           print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

6.2 Diabetes Dataset

Importing the Dependencies

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import classification_report,confusion_matrix, ConfusionMatrixDisplay
        import matplotlib.pyplot as plt
```

Data Collection

```
In [2]: # loading the diabetes dataset to a pandas DataFrame
        diabetes_dataset = pd.read_csv('/home/srivathsa/Downloads/Mini Project/data/diabetes.csv')
```

```
In [3]: # printing the first 5 rows of the dataset
        diabetes_dataset.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [4]: # number of rows and Columns in this dataset
        diabetes_dataset.shape

Out[4]: (768, 9)
```

```
In [5]: # getting the statistical measures of the data
        diabetes_dataset.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
In [21]: diabetes_dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Train Test Split

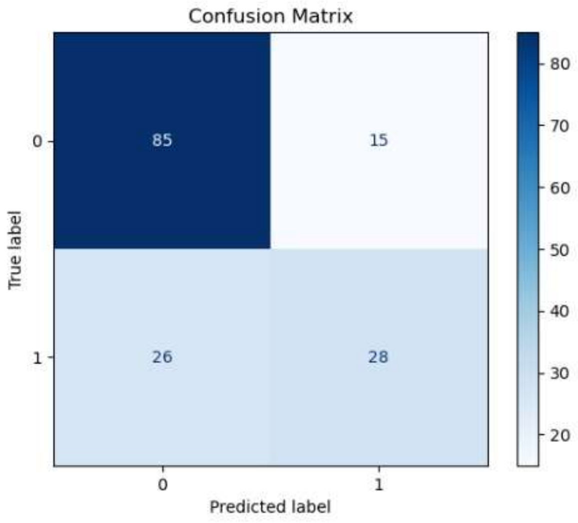
```
In [11]: X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)

In [12]: print(X.shape, X_train.shape, X_test.shape)

(768, 8) (614, 8) (154, 8)
```

Training the Model

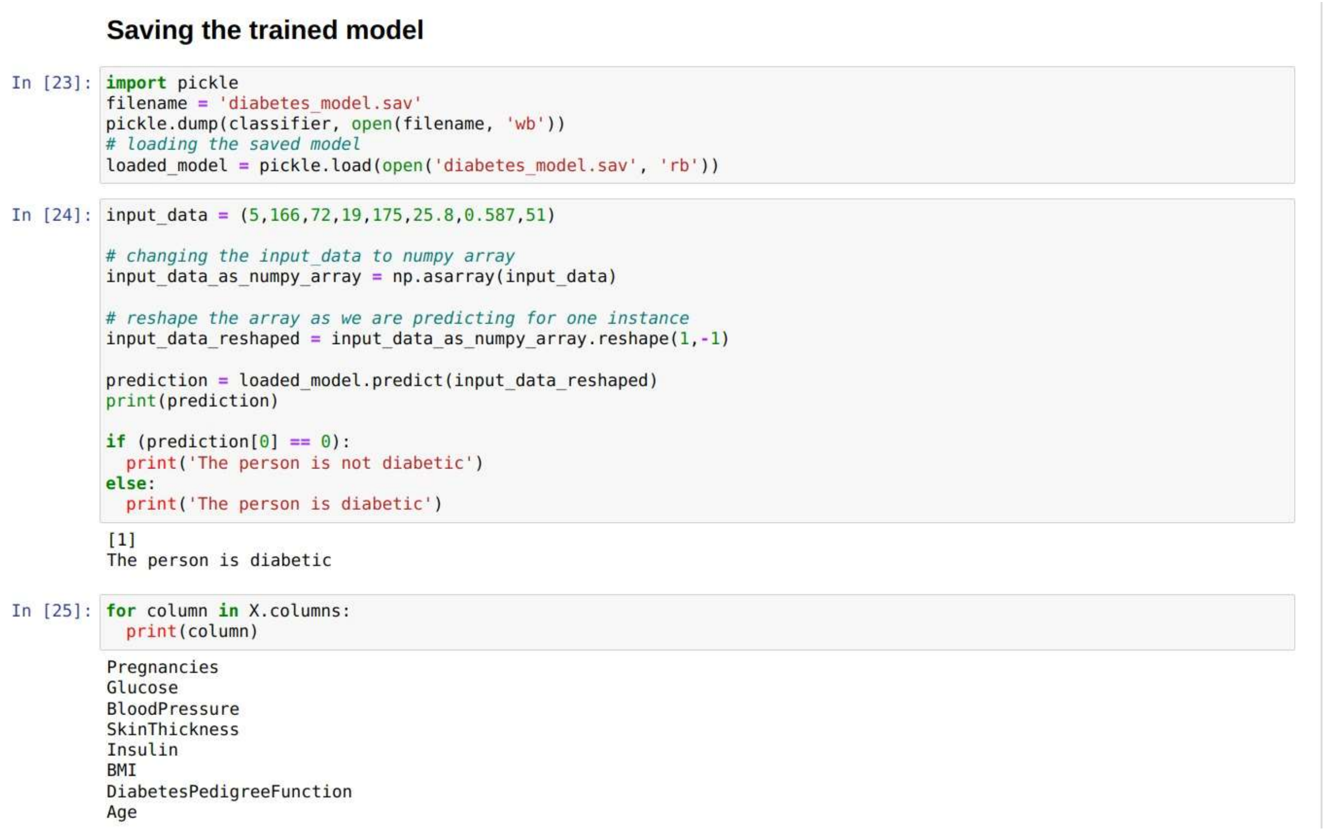
```
In [13]: classifier = RandomForestClassifier()

In [14]: #training the support vector Machine Classifier
         classifier.fit(X_train, Y_train)

Out[14]:
▾ RandomForestClassifier
RandomForestClassifier()
```

Model Evaluation

```
In [17]: classifier_1 = RandomForestClassifier(n_estimators = 160).fit(X_train,Y_train)
         y_pred_1 = classifier_1.predict(X_test)
         cm = confusion_matrix(Y_test, y_pred_1)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm)
         disp.plot(cmap='Blues')
         plt.title('Confusion Matrix')
         plt.show()
         print(classification_report(Y_test,y_pred_1))
```



```
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       100
           1       0.65      0.52      0.58        54

    accuracy                           0.73       154
   macro avg       0.71      0.68      0.69       154
weighted avg       0.73      0.73      0.73       154
```



Saving the trained model

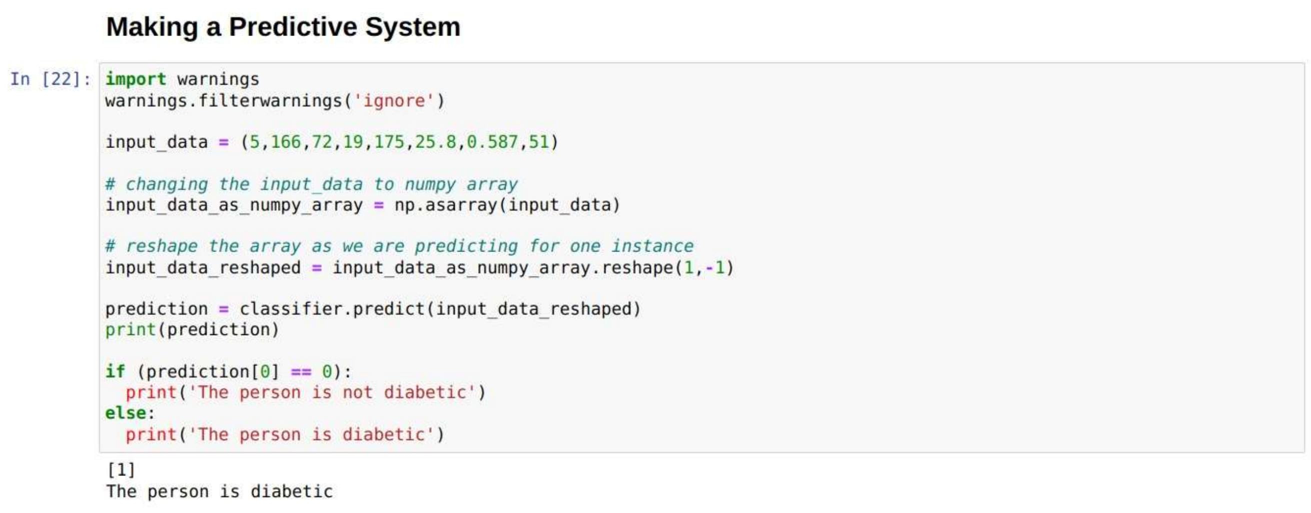
In [23]:
```
import pickle
filename = 'diabetes_model.sav'
pickle.dump(classifier, open(filename, 'wb'))
# loading the saved model
loaded_model = pickle.load(open('diabetes_model.sav', 'rb'))
```

In [24]:
```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[1]
The person is diabetic
```

In [25]:
```
for column in X.columns:
  print(column)
```

```
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
```

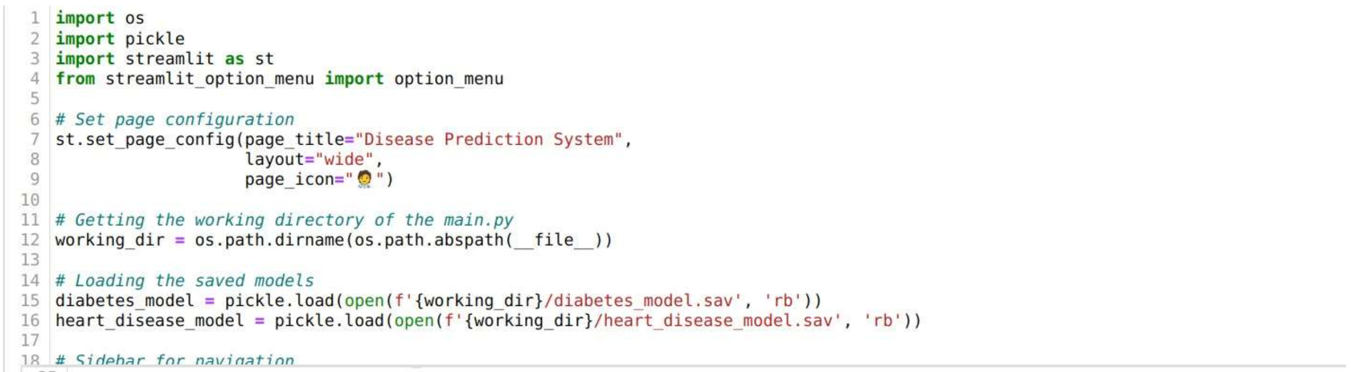


Making a Predictive System

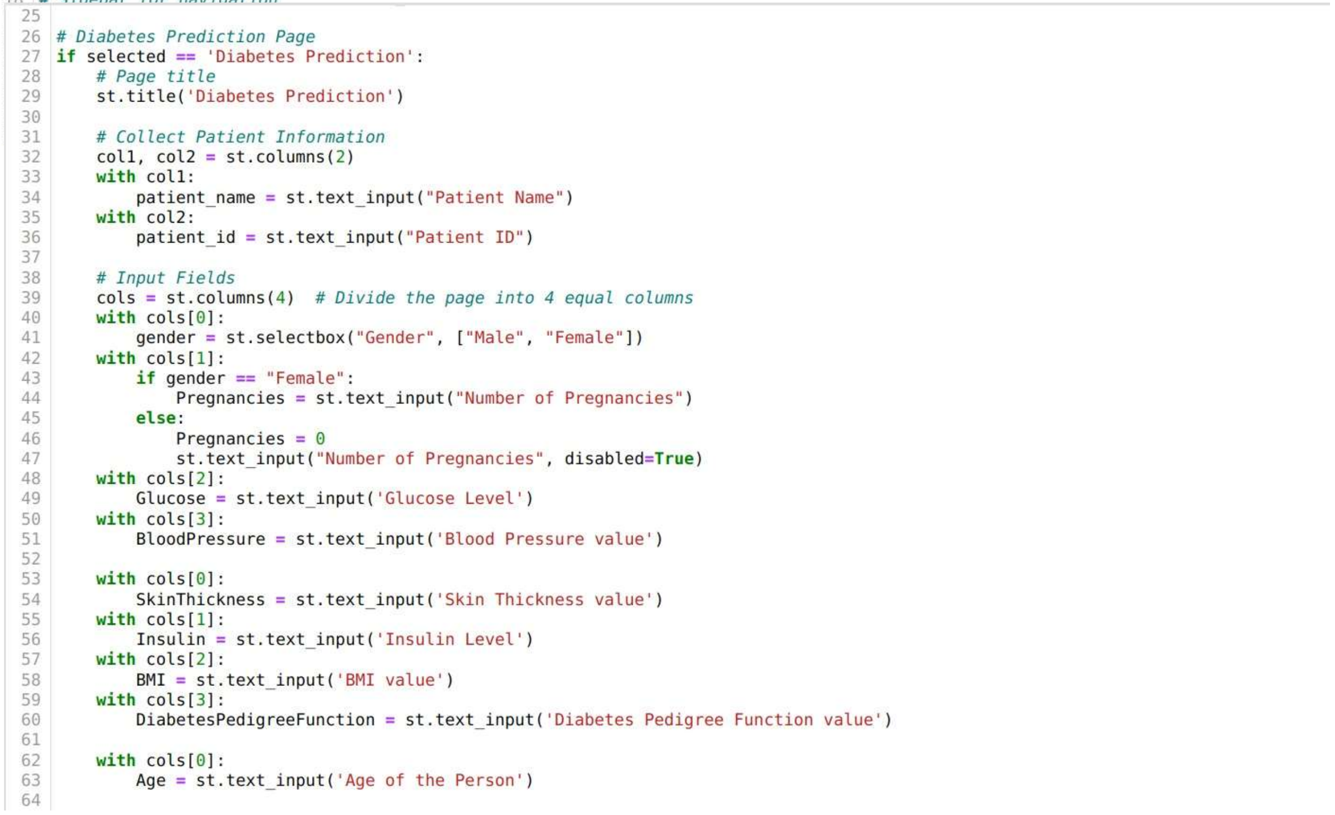
In [22]:
```
import warnings
warnings.filterwarnings('ignore')

input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = classifier.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[1]
The person is diabetic
```

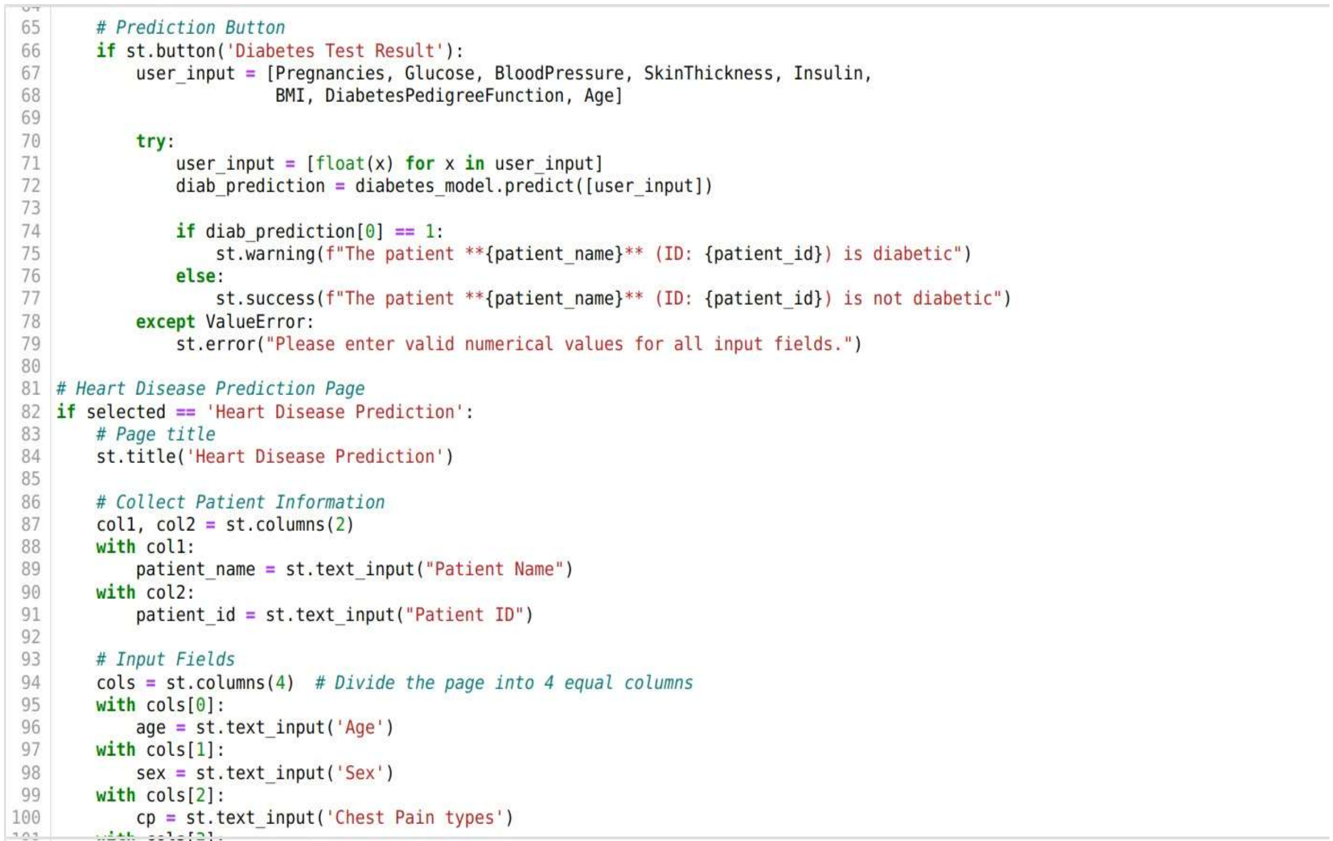
6.3 PREDICTIVE SYSTEM (USER INTERFACE):



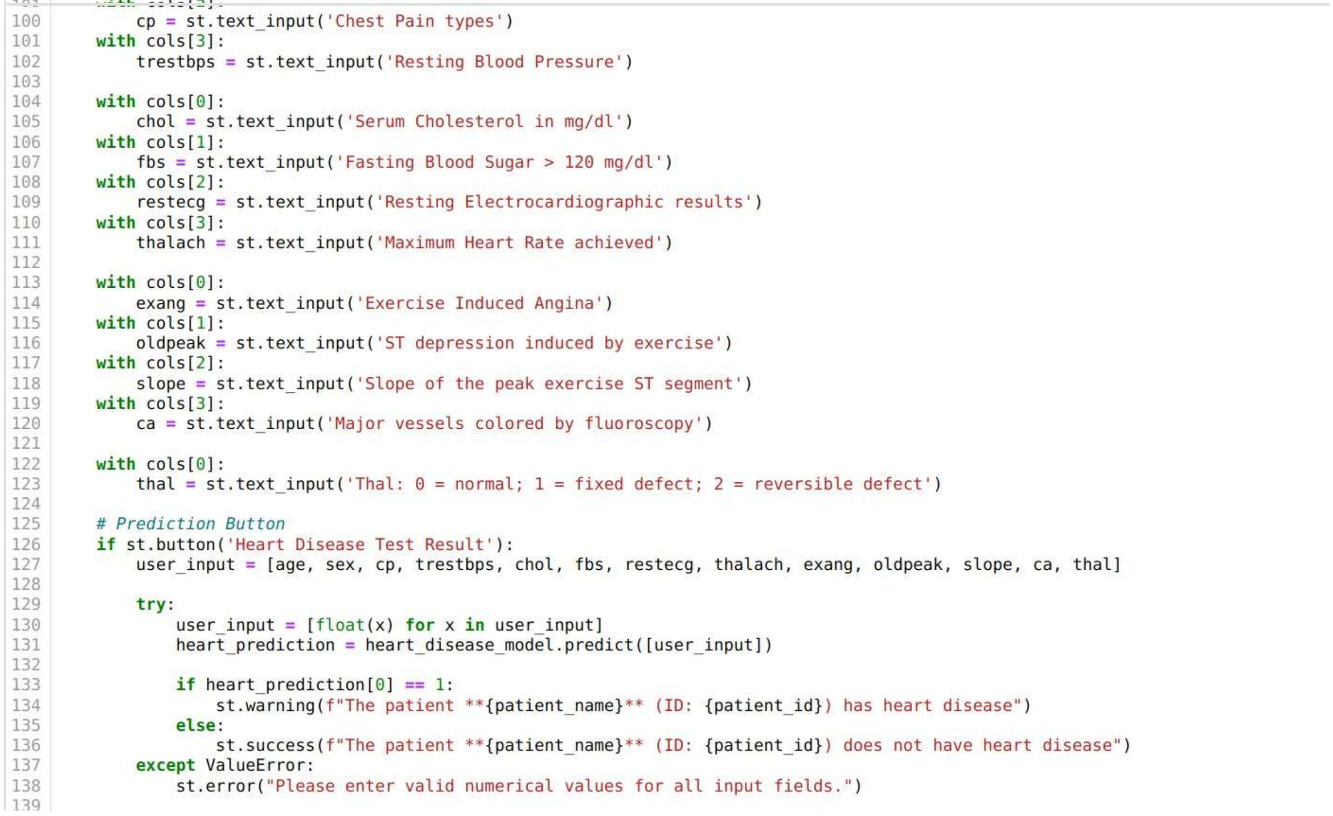
```python
import os
import pickle
import streamlit as st
from streamlit_option_menu import option_menu

# Set page configuration
st.set_page_config(page_title="Disease Prediction System",
                   layout="wide",
                   page_icon="🧑‍⚕️")

# Getting the working directory of the main.py
working_dir = os.path.dirname(os.path.abspath(__file__))

# Loading the saved models
diabetes_model = pickle.load(open(f'{working_dir}/diabetes_model.sav', 'rb'))
heart_disease_model = pickle.load(open(f'{working_dir}/heart_disease_model.sav', 'rb'))

# Sidebar for navigation
```



```python

# Diabetes Prediction Page
if selected == 'Diabetes Prediction':
    # Page title
    st.title('Diabetes Prediction')

    # Collect Patient Information
    col1, col2 = st.columns(2)
    with col1:
        patient_name = st.text_input("Patient Name")
    with col2:
        patient_id = st.text_input("Patient ID")

    # Input Fields
    cols = st.columns(4)  # Divide the page into 4 equal columns
    with cols[0]:
        gender = st.selectbox("Gender", ["Male", "Female"])
    with cols[1]:
        if gender == "Female":
            Pregnancies = st.text_input("Number of Pregnancies")
        else:
            Pregnancies = 0
            st.text_input("Number of Pregnancies", disabled=True)
    with cols[2]:
        Glucose = st.text_input('Glucose Level')
    with cols[3]:
        BloodPressure = st.text_input('Blood Pressure value')

    with cols[0]:
        SkinThickness = st.text_input('Skin Thickness value')
    with cols[1]:
        Insulin = st.text_input('Insulin Level')
    with cols[2]:
        BMI = st.text_input('BMI value')
    with cols[3]:
        DiabetesPedigreeFunction = st.text_input('Diabetes Pedigree Function value')

    with cols[0]:
        Age = st.text_input('Age of the Person')

```

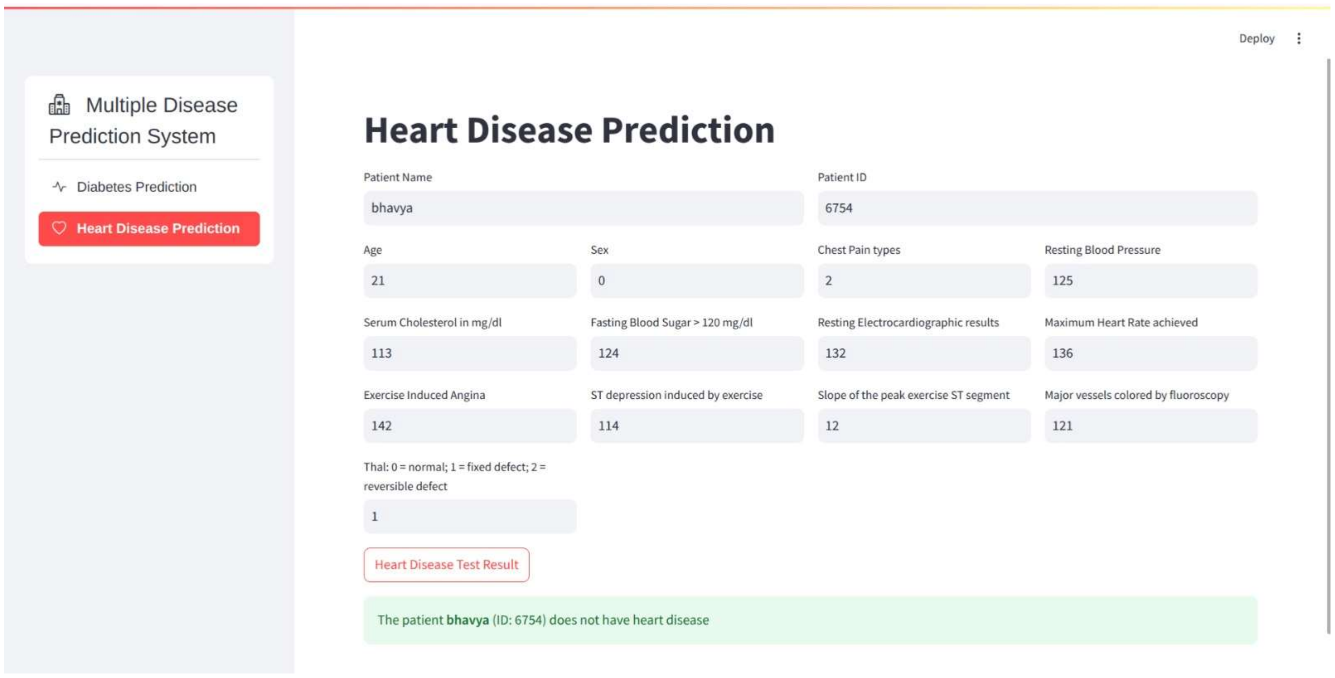


```python
    # Prediction Button
    if st.button('Diabetes Test Result'):
        user_input = [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin,
                      BMI, DiabetesPedigreeFunction, Age]

        try:
            user_input = [float(x) for x in user_input]
            diab_prediction = diabetes_model.predict([user_input])

            if diab_prediction[0] == 1:
                st.warning(f"The patient **{patient_name}** (ID: {patient_id}) is diabetic")
            else:
                st.success(f"The patient **{patient_name}** (ID: {patient_id}) is not diabetic")
        except ValueError:
            st.error("Please enter valid numerical values for all input fields.")

# Heart Disease Prediction Page
if selected == 'Heart Disease Prediction':
    # Page title
    st.title('Heart Disease Prediction')

    # Collect Patient Information
    col1, col2 = st.columns(2)
    with col1:
        patient_name = st.text_input("Patient Name")
    with col2:
        patient_id = st.text_input("Patient ID")

    # Input Fields
    cols = st.columns(4)  # Divide the page into 4 equal columns
    with cols[0]:
        age = st.text_input('Age')
    with cols[1]:
        sex = st.text_input('Sex')
    with cols[2]:
        cp = st.text_input('Chest Pain types')
```



```python
        cp = st.text_input('Chest Pain types')
    with cols[3]:
        trestbps = st.text_input('Resting Blood Pressure')

    with cols[0]:
        chol = st.text_input('Serum Cholesterol in mg/dl')
    with cols[1]:
        fbs = st.text_input('Fasting Blood Sugar > 120 mg/dl')
    with cols[2]:
        restecg = st.text_input('Resting Electrocardiographic results')
    with cols[3]:
        thalach = st.text_input('Maximum Heart Rate achieved')

    with cols[0]:
        exang = st.text_input('Exercise Induced Angina')
    with cols[1]:
        oldpeak = st.text_input('ST depression induced by exercise')
    with cols[2]:
        slope = st.text_input('Slope of the peak exercise ST segment')
    with cols[3]:
        ca = st.text_input('Major vessels colored by fluoroscopy')

    with cols[0]:
        thal = st.text_input('Thal: 0 = normal; 1 = fixed defect; 2 = reversible defect')

    # Prediction Button
    if st.button('Heart Disease Test Result'):
        user_input = [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]

        try:
            user_input = [float(x) for x in user_input]
            heart_prediction = heart_disease_model.predict([user_input])

            if heart_prediction[0] == 1:
                st.warning(f"The patient **{patient_name}** (ID: {patient_id}) has heart disease")
            else:
                st.success(f"The patient **{patient_name}** (ID: {patient_id}) does not have heart disease")
        except ValueError:
            st.error("Please enter valid numerical values for all input fields.")
```

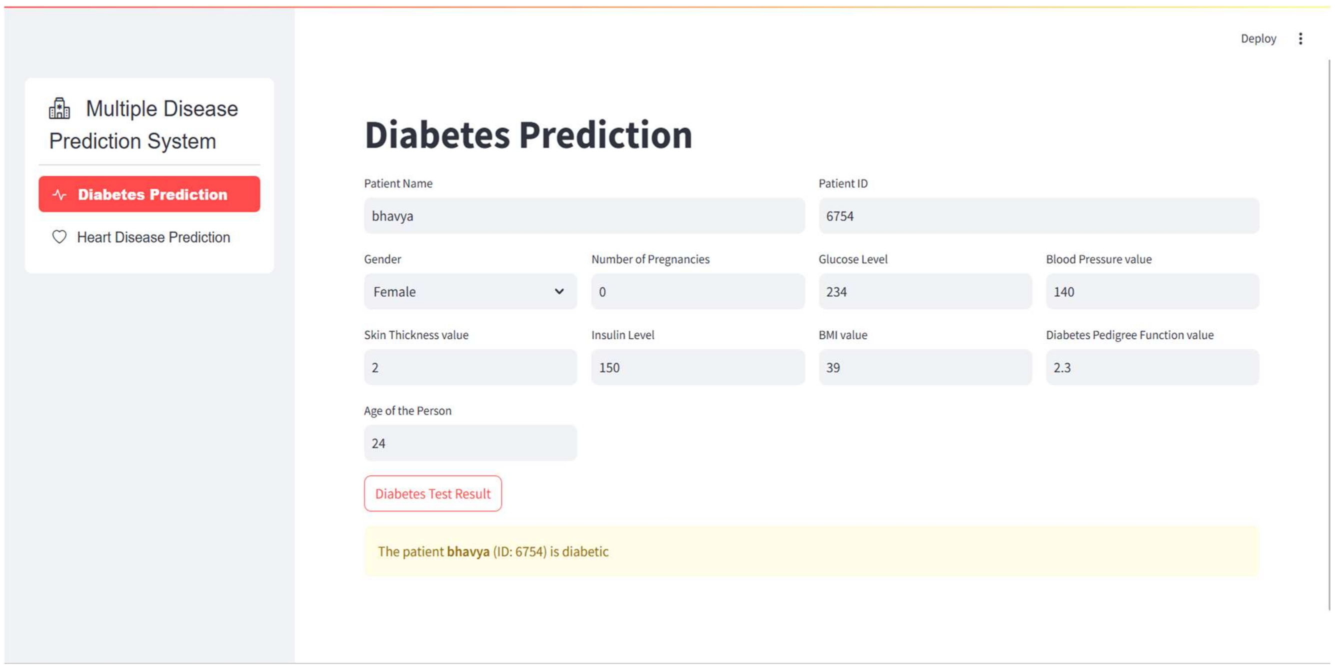
OUTPUT:



Heart Disease Prediction

Patient Name	bhavya
Patient ID	6754
Age	21
Sex	0
Chest Pain types	2
Resting Blood Pressure	125
Serum Cholestrol in mg/dl	113
Fasting Blood Sugar > 120 mg/dl	124
Resting Electrocardiographic results	132
Maximum Heart Rate achieved	136
Exercise Induced Angina	142
ST depression induced by exercise	114
Slope of the peak exercise ST segment	12
Major vessels colored by flourosopy	121
Thal: 0 = normal; 1 = fixed defect; 2 = reversible defect	1

The patient **bhavya** (ID: 6754) does not have heart disease



Diabetes Prediction

Patient Name	bhavya
Patient ID	6754
Gender	Female
Number of Pregnancies	0
Glucose Level	234
Blood Pressure value	140
Skin Thickness value	2
Insulin Level	150
BMI value	39
Diabetes Pedigree Function value	2.3
Age of the Person	24

The patient **bhavya** (ID: 6754) is diabetic

7. DISCUSSION

7.1 Interpretation of the results in the context of problem:

The results of the "Heart Disease and Diabetes Prediction" project demonstrate the system's effectiveness in accurately predicting the likelihood of these diseases based on clinical data and risk factors. The high accuracy and precision scores indicate a reliable prediction process. The system exhibits commendable recall, identifying significant risk factors with low latency for real-time decision-making. User feedback emphasizes the system's usability and practical application, providing valuable insights for improvements. Despite successes, data limitations and algorithmic challenges are acknowledged, impacting prediction accuracy in certain scenarios. Practical applicability is affirmed through meaningful and actionable insights, contributing to improved healthcare interventions.

Identified weaknesses include potential biases and a need for diverse and comprehensive datasets. Suggestions for enhancement encompass expanding the dataset to include a broader range of demographic and clinical data and incorporating advanced machine learning algorithms to enhance prediction accuracy. Ethical considerations highlight the importance of fairness, transparency, and accessibility, ensuring equitable healthcare support and reliable predictions for diverse populations.

In conclusion, the system successfully addresses the problem, meeting project objectives. The interpretation underscores achievements, acknowledges limitations, and outlines a roadmap for future enhancements, ensuring a significant and meaningful impact on healthcare outcomes for the target audience.

7.2 Strengths and weaknesses:

Strengths:

- **High Accuracy**: The system achieves reliable predictions, ensuring accurate identification of the likelihood of heart disease and diabetes.

- **Real-time Predictions**: Low latency enhances practical usability, making the system suitable for timely healthcare decision-making.
- **User-Friendly Interface:** The system is easy to use, helping patients manage heart disease or diabetes effectively.
- **Contextually Relevant Gestures:** The system is easy to use, helping patients manage their health and It provides clear guidance and supports different accents for inclusivity.

Weaknesses:

- **Limited Data**: A narrow dataset may affect the system’s ability to recognize diverse symptoms or variations
- **Algorithmic Limitations**: Complex cases might challenge the performance of underlying algorithms.
- **Potential Biases**: Biases in health data analysis could lead to disparities in care recommendations.
- **Dependence on Accuracy**: The system may rely on accurate inputs, creating challenges with incomplete or unclear information.
- **Privacy Concerns**: Managing sensitive patient data raises ethical concerns about security and consent.
- **Language Constraints**: Limited language support might exclude some patients from accessing the system effectively.

Overall Assessment:

While the strengths highlight the system's positive impact on patient care and satisfaction, addressing weaknesses, such as data limitations and biases, is essential for improvement. By resolving these issues, the system can better support patients in managing heart disease and diabetes effectively and inclusively.

8. CONCLUSION

This project demonstrates the transformative potential of machine learning (ML) in addressing significant global health challenges, specifically heart disease and diabetes. These conditions are among the leading causes of mortality and complications worldwide, necessitating early detection and proactive management.

By leveraging shared risk factors such as glucose levels, cholesterol, BMI, and age, the system provides a unified predictive model. Unlike traditional methods, the proposed approach integrates advanced ML techniques and real-time data, delivering superior accuracy, scalability, and adaptability.

The project pipeline incorporates several essential steps to ensure optimal model performance. Data collection gathers patient records, medical histories, and real-time metrics from wearable devices, creating a comprehensive dataset. Preprocessing ensures data quality by addressing missing values, outliers, and inconsistencies while normalizing features for uniformity.

Feature engineering derives meaningful variables like BMI and risk scores, emphasizing the most relevant predictors through correlation analysis and domain expertise. Hyperparameter tuning, conducted using grid search and cross-validation, optimizes the model's performance and enhances its reliability. This systematic approach ensures that the model is robust and applicable in real-world healthcare settings.

Unlike traditional systems, which often rely on static models, this system dynamically adapts to new data, reflecting real-time trends and changes in patient health. Integration with wearable devices allows for continuous monitoring, enabling healthcare professionals to access up-to-date risk assessments.

The system is deployed as a web application using Streamlit, offering an intuitive interface for inputting patient data and accessing visualized predictions. This user-friendly design empowers users to make informed decisions about early diagnosis, preventive measures, and treatment plans.

Decision Trees, the proposed system addresses critical limitations. Traditional models often assume linear relationships or struggle with scalability and high-dimensional data.

For example, Logistic Regression cannot capture complex interactions, and SVMs, though effective with non-linear data, are computationally intensive and lack scalability for large datasets. In contrast, Random Forest and Neural Networks excel in handling complex data patterns and providing improved accuracy, robustness, and scalability. Additionally, the feature importance scores generated by Random Forest enhance interpretability, a key factor in clinical adoption.

While the system addresses significant limitations of existing methods, there are areas for improvement and future expansion. One such area is interpretability, which can be enhanced using explainable AI techniques like SHAP (SHapley Additive exPlanations). These methods can make the system's predictions more transparent and understandable for healthcare professionals.

Expanding the model to predict other chronic conditions, such as hypertension or kidney disease, could broaden its applicability. Incorporating advanced time-series models, such as Long Short-Term Memory (LSTM) networks, would enable dynamic risk prediction based on sequential data. Additionally, integrating IoT-based monitoring devices would further enhance real-time data updates and improve patient management.

In conclusion, this project highlights the immense potential of ML in revolutionizing healthcare diagnostics and management. Through its robust algorithms, real-time adaptability, and user-friendly deployment, the system provides a scalable and accurate solution for predicting heart disease and diabetes. With continuous refinement, such as enhanced interpretability and expanded capabilities, the system has the potential to become a critical tool in modern healthcare.

9. FUTURE WORKS

The proposed system for predicting heart disease and diabetes is a major step forward, but there are many ways it can be expanded and improved to make it even more effective and accessible. One of the first areas to focus on is making the system easier to understand for both doctors and patients. Machine learning (ML) models like Random Forest and Neural Networks are highly accurate but often function as "black boxes," meaning it’s hard to explain how they make decisions.

To solve this, explainable AI (XAI) techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) can be added. These tools can break down the predictions and show which factors, such as glucose levels, cholesterol, or BMI, played the biggest role. This transparency helps build trust and allows doctors to use the system confidently as part of their clinical decision-making process.

Another exciting future direction is to expand the system beyond heart disease and diabetes. Many chronic diseases share risk factors, including obesity, high blood pressure, and a sedentary lifestyle. By integrating these shared factors, the system could also predict conditions like kidney disease, hypertension, or even some types of cancer.

This would create a comprehensive multi-disease prediction platform that helps healthcare providers address multiple risks at once, improving patient outcomes. For example, if a patient is at risk for both diabetes and hypertension, the system could provide tailored recommendations for managing both conditions, ensuring better overall health management.

NLP tools can extract useful insights from these sources, such as identifying symptoms mentioned in a doctor's notes or understanding patient sentiment from their feedback. By adding these qualitative insights, the system could provide more accurate and personalized predictions.

Accessibility is another key area for improvement. While the current system is deployed as a web application, creating a mobile app would make it even more accessible for patients and healthcare providers. A mobile app could allow patients to input their health data, such as daily glucose readings or exercise habits, directly from their smartphones. It could also provide instant feedback on their risk levels and offer personalized health tips.

Finally, the system should undergo real-world testing through clinical trials to validate its effectiveness and reliability. Clinical trials would allow researchers to evaluate the system's impact on patient outcomes, such as reducing complications or improving disease management. These trials could also help identify areas for further refinement, ensuring the system meets the needs of both patients and healthcare providers. For example, testing the system in hospitals or clinics could reveal how well it integrates into existing workflows and whether it reduces the workload for medical staff.

In conclusion, the future of this system is full of opportunities for innovation and improvement. By enhancing interpretability, expanding its scope to include more diseases, and integrating real-time data from wearable devices, the system can become even more valuable for healthcare.

10. REFERENCES

1. Katarya, R., & Srinivas, P. (2020). Predicting heart disease at early stages using machine learning: A survey. *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 302–305.
2. Parashar, A., Gupta, A., & Gupta, A. (2014). Machine learning techniques for diabetes prediction. *International Journal of Emerging Technologies and Advanced Engineering*, 4(3), 672–675.
3. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
4. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.
5. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?" Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.
6. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.
7. Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations (ICLR)*.
8. Rajkomar, A., Dean, J., & Kohane, I. (2019). Machine Learning in Medicine. *New England Journal of Medicine*, 380(14), 1347–1358.

9. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

10. Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61, 85–117.

11. Kermany, D. S., Goldbaum, M., et al. (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, 172(5), 1122–1131.

12. Google Health. (2021). Federated Learning: Privacy-Preserving AI for Healthcare. Retrieved from https://health.google.

13. Shickel, B., Tighe, P. J., Bihorac, A., & Rashidi, P. (2018). Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis. *Journal of Biomedical Informatics*, 83, 168–185.